```
ISIS-II PL/M-80 V3.1 COMPILATION OF MODULE TYPE
OBJECT MODULE PLACED IN TYPE.OBJ
COMPILER INVOKED BY:  PLM80 TYPE.PLM PAGEWIDTH(100) DEBUG OPTIMIZE
```

```
          $ TITLE('CP/M 3.0 --- TYPE ')
  1       type:
          do;

          /*
            Copyright (C) 1982
            Digital Research
            P.O. Box 579
            Pacific Grove, CA 93950
          */

          /*
            Revised:
              19 Jan  80  by Thomas Rolander
              14 Sept 81  by Doug Huskey
              07 July 82  by John Knight
              06 Oct  82  by Doug Huskey
              02 Dec  82  by Bruce Skidmore
          */
```

CP/M - PLUS V 3.0
COPYRIGHT © 1982
DIGITAL RESEARCH
P. O. BOX 579
PACIFIC GROVE, CA 93950
SER. # CP3-135

```
  2  1    declare
              mpmproduct literally '01h', /* requires mp/m */
              cpmversion literally '30h'; /* requires 3.0 cp/m */


            /****************************************
             *                                      *
             *       EQUATES (LITERALS)             *
             *                                      *
             ****************************************/


  3  1    declare
              true          literally '0FFh',
              false         literally '0',
              forever       literally 'while true',
              lit           literally 'literally',
              proc          literally 'procedure',
              dcl           literally 'declare',
              addr          literally 'address',
              cr            literally '13',
              lf            literally '10',
              ctrli         literally '9',
              ctrlc         literally '3',
              ctrlo         literally '0fh',
              ctrlx         literally '18h',
              bksp          literally '8',
              dcnt$offset   literally '45h',
              searcha$offset literally '47h',
              searchl$offset literally '49h',
              hashl$offset  literally '00h',
```

```
                hash2$offset   literally '02h',
                hash3$offset   literally '04h',
                con$page$mode  literally '2ch',
                con$page$size  literally '1ch';

               /*****************************************
                *                                       *
                *        GLOBAL VARIABLES               *
                *                                       *
                *****************************************/

 4   1      declare plm label public;
 5   1      declare (eod,i,char) byte;
 6   1      declare control$z literally '1AH';
 7   1      declare (cnt,tcnt,code) byte;
 8   1      declare (ver, error$code)  address;
 9   1      declare paging byte initial (true);
10   1      declare negate byte initial (false);
11   1      declare status address;
12   1      declare m based status byte;
13   1      declare no$chars byte;
14   1      declare last$dseg$byte byte initial (0);
15   1      declare wflag byte initial (false);
16   1      declare cur$fcb (33) byte;      /* current fcb (to type) */

17   1      declare
              more    (*)  byte data (cr,lf,cr,lf,'Press RETURN to Continue $'),
              failed  (*)  byte data(cr,lf,'ERROR: Not typed: $');



               /*****************************************
                *                                       *
                *      B D O S   INTERFACE              *
                *                                       *
                *****************************************/


18   1      mon1:
              procedure (func,info) external;
19   2          declare func byte;
20   2          declare info address;
21   2        end mon1;

22   1      mon2:
              procedure (func,info) byte external;
23   2          declare func byte;
24   2          declare info address;
25   2        end mon2;

26   1      mon3:
              procedure (func,info) address external;
27   2          declare func byte;
28   2          declare info address;
29   2        end mon3;

30   1      declare cmdrv     byte    external;   /* command drive      */
```

COPYRIGHT © 1982
DIGITAL RESEARCH
P. O. BOX 579
PACIFIC GROVE, CA 93950
SER. #_____

```
 31  1     declare fcb (1)   byte    external;  /* 1st default fcb    */
 32  1     declare fcb16 (1) byte    external;  /* 2nd default fcb    */
 33  1     declare pass0     address external;  /* 1st password ptr   */
 34  1     declare len0      byte    external;  /* 1st passwd length  */
 35  1     declare pass1     address external;  /* 2nd password ptr   */
 36  1     declare len1      byte    external;  /* 2nd passwd length  */
 37  1     declare tbuff (1) byte    external;  /* default dma buffer */


           /**************************************
            *                                    *
            *      B D O S   Externals           *
            *                                    *
            **************************************/

 38  1     read$console:
          procedure byte;
 39  2       return mon2 (1,0);
 40  2       end read$console;

 41  1     printchar:
          procedure (char);
 42  2       declare char byte;
 43  2       call mon1 (2,char);
 44  2       end printchar;

 45  1     conin:
          procedure byte;
 46  2       return mon2(6,0fdh);
 47  2       end conin;

 48  1     print$buf:
          procedure (buff$adr);
 49  2       declare buff$adr address;
 50  2       call mon1 (9,buff$adr);
 51  2       end print$buf;

 52  1     read$console$buf:
          procedure (buffer$address,max) byte;
 53  2       declare buffer$address address;
 54  2       declare new$max based buffer$address address;
 55  2       declare max byte;
 56  2       new$max = max;
 57  2       call mon1(10,buffer$address);
 58  2       buffer$address = buffer$address + 1;
 59  2       return new$max;   /* actually number of chars input */
 60  2     end read$console$buf;

 61  1     version: procedure address;
          /* returns current cp/m version # */
 62  2       return mon3(12,0);
 63  2       end version;

 64  1     check$con$stat:
          procedure byte;
 65  2       return mon2 (11,0);
 66  2       end check$con$stat;
```

COPYRIGHT © 1982
DIGITAL RESEARCH
P. O. BOX 579
PACIFIC GROVE, CA 93950
SER. #_____

```
 67  1      open$file:
              procedure (fcb$address) address;
 68  2            declare fcb$address address;
 69  2            return mon3(15,fcb$address);
 70  2            end open$file;

 71  1      close$file:
              procedure (fcb$address) byte;
 72  2            declare fcb$address address;
 73  2            return mon2 (16,fcb$address);
 74  2          end close$file;

 75  1      read$record:
              procedure (fcb$address) byte;
 76  2            declare fcb$address address;
 77  2            return mon2 (20,fcb$address);
 78  2          end read$record;

 79  1      setdma: procedure(dma);
 80  2          declare dma address;
 81  2          call mon1(26,dma);
 82  2          end setdma;

             /* Off & Ofe = return BDOS errors */
 83  1      return$errors:
              procedure(mode);
 84  2          declare mode byte;
 85  2            call mon1 (45,mode);
 86  2          end return$errors;

 87  1      terminate:
              procedure;
 88  2            call mon1 (0,0);
 89  2          end terminate;

 90  1      search$first:
              procedure (fcb$address) byte;
 91  2            declare fcb$address address;
 92  2            return mon2 (17,fcb$address);
 93  2          end search$first;

 94  1      search$next:
              procedure byte;
 95  2            return mon2 (18,0);
 96  2          end search$next;

 97  1      declare scbpd structure
                (offset byte,
                 set    byte,
                 value  address);

 98  1      getscbbyte:
              procedure (offset) byte;
 99  2            declare offset byte;
100  2            scbpd.offset = offset;
```

COPYRIGHT © 1982
DIGITAL RESEARCH
P. O. BOX 579
PACIFIC GROVE, CA 93950
SER. #_____

```
101  2            scbpd.set = 0;
102  2            return mon2(49,.scbpd);
103  2          end getscbbyte;

104  1        getscbword:
               procedure (offset) address;
105  2          declare offset byte;
106  2          scbpd.offset = offset;
107  2          scbpd.set = 0;
108  2          return mon3(49,.scbpd);
109  2        end getscbword;

110  1        setscbword:
               procedure (offset,value);
111  2            declare offset byte;
112  2            declare value address;
113  2            scbpd.offset = offset;
114  2            scbpd.set = 0FEh;
115  2            scbpd.value = value;
116  2            call mon1(49,.scbpd);
117  2        end setscbword;

118  1        set$console$mode: procedure;
             /* set console mode to control-c only */
119  2          call mon1(109,1);
120  2        end set$console$mode;

121  1        declare
             parse$fn structure (
               buff$adr  address,
               fcb$adr   address);

122  1        parse: procedure(pfcb) address external;
123  2          declare pfcb address;
124  2          end parse;


             /****************************************
              *                                      *
              *      S U B R O U T I N E S           *
              *                                      *
              ****************************************/

                          /* upper case character from console */
125  1      crlf:   proc;
126  2          call printchar(cr);
127  2          call printchar(lf);
128  2          end crlf;
             /* ------------------------------------ */


                          /* fill string @ s for c bytes with f */
129  1      fill:   proc(s,f,c);
130  2          dcl s addr,
                   (f,c) byte,
                   a based s byte;
```

COPYRIGHT © 1982
DIGITAL RESEARCH
P. O. BOX 579
PACIFIC GROVE, CA 93950

SER. #_____

```
131  2           do while (c:=c-1)<>255;
132  3           a = f;
133  3           s = s+1;
134  3           end;
135  2       end fill;

        /* - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - */

                        /* upper case character from console */
136  1   ucase:  proc byte;
137  2       dcl c byte;

138  2       if (c:=conin) >= 'a' then
139  2          if c < '{' then
140  2             return(c-20h);
141  2       return c;
142  2       end ucase;

        /* - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - */

                        /* get password and place at fcb + 16 */
143  1   getpasswd:  proc;
144  2       dcl (i,c) byte;

145  2       call crlf;
146  2       call crlf;
147  2       call print$buf(.('Password: ','$'));
148  2   retry:
           call fill(.fcb16,' ',8);
149  2          do i = 0 to 7;
150  3   nxtchr:
              if (c:=ucase) >= ' ' then
151  3                fcb16(i)=c;
152  3             if c = cr then
153  3                go to exit;
154  3             if c = ctrlx then
155  3                goto retry;
156  3             if c = bksp then do;
158  4                if i<1 then
159  4                   goto retry;
160  4                else do;
161  5                   fcb16(i:=i-1)=' ';
162  5                   goto nxtchr;
163  5                   end;
164  4                end;
165  3             if c = 3 then
166  3                call terminate;
167  3             end;
168  2   exit:
          c = check$con$stat;          /* clear raw I/O mode */
169  2       end getpasswd;

        /* - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - */

                        /* error message routine */
170  1   error:  proc(code);
171  2     declare
```

COPYRIGHT © 1982
DIGITAL RESEARCH
P. O. BOX 579
PACIFIC GROVE, CA 93950
SER. #

```
                code byte;

172  2        if code=0 then do;
174  3           call print$buf (.('No File','$'));
175  3           call terminate;
176  3           end;
177  2        if code=1 then do;
179  3           call print$buf(.(cr,lf,'BDOS Bad Sector$'));
180  3           call terminate;
181  3           end;
182  2        if code=4 then do;
184  3           call print$buf(.(cr,lf,'Invalid Drive$'));
185  3           call terminate;
186  3           end;
187  2        if code = 5 then
188  2           call print$buf(.('Currently Opened$ '));
189  2        if code = 7 then
190  2           call print$buf(.('Password Error$'));
191  2        end error;

          /* - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - */

                     /* print file name */
192  1     print$file: procedure(fcbp);
193  2        declare (K,c) byte;
194  2        declare typ lit '9';        /* file type */
195  2        declare fnam lit '11';       /* file type */
196  2        declare
                    fcbp   addr,
                    fcbv   based fcbp (32) byte;

197  2        if fcbv(0) <> 0 then do;
199  3           call printchar(fcbv(0)+'@');
200  3           call printchar(':');
201  3        end;

202  2           do K = 1 to fnam;
203  3           if K = typ then
204  3             call printchar('.');
205  3           if (c := (fcbv(K) and 7fh)) <> ' ' then
206  3             call printchar(c);
207  3           end;

208  2     end print$file;

          /* - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - */

209  1     error$opt: procedure (code);
210  2        declare code byte;
211  2        call print$buf(.('ERROR: $'));
212  2        if code = 0 then
213  2           call print$buf(.('Invalid or missing delimiter(s) $'));
214  2        if code = 1 then
215  2           call print$buf(.('Try ''PAGE'' or ''NO PAGE'' $'));
216  2        call terminate;
217  2     end error$opt;
```

COPYRIGHT © 1982
DIGITAL RESEARCH
P. O. BOX 579
PACIFIC GROVE, CA 93950
SER. #

```
                /* - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - */

218   1         input$found: procedure (buffer$adr) byte;
219   2           declare buffer$adr address;
220   2           declare char based buffer$adr byte;
221   2           do while (char = ' ') or (char = 9); /* tabs & spaces */
222   3             buffer$adr = buffer$adr + 1;
223   3           end;
224   2           if char = 0 then  /* eoln */
225   2             return false;   /* input not found */
                  else
226   2             return true;    /* input found */
227   2         end input$found;

                /* - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - */

228   1         scanques: procedure(str$addr) byte;
229   2            declare str$addr address;
230   2            declare char based str$addr byte;
231   2            declare i byte;
232   2            declare wildcard byte;

233   2            i = 0;
234   2            wildcard = false;
235   2            do while (i < 11);
236   3               if char = '?' then
237   3                  wildcard = true;
238   3               i = i + 1;
239   3               str$addr = str$addr + 1;
240   3               end; /* do while */
241   2            return wildcard;
242   2         end scanques;

                /*- - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -*/

                                /* skip over blanks or tabs in command */
243   1         page$test: procedure;

244   2             if cnt <> 0 then
245   2               if (tcnt:=tcnt+1) >= cnt then do;
247   3                 call print$buf(.more);
248   3                 tcnt = conin;
249   3                 call print$char(cr);
250   3                 if tcnt = ctrlc then
251   3                   call terminate;
252   3                 if tcnt = ctrlo then
253   3                   eod = true;
254   3                 tcnt = -1;
255   3               end;
256   2         end page$test;

                /* - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - */

                                /* type a file specified by FCB */
257   1         type$file: procedure;

258   2             call return$errors(0FFh);          /* return after error message */
```

COPYRIGHT © 1982
DIGITAL RESEARCH
P. O. BOX 579
PACIFIC GROVE, CA 93950
SER. #

```
259  2          call setdma(.fcb16);              /* set dma to password */
260  2          curfcb(6) = curfcb(6) or 80h;           /* open in RO mode     */
261  2          curfcb(12) = 0;              /* open zero extent    */
262  2          error$code = open$file (.curfcb);
263  2          if low(error$code) = 0FFh then
264  2            if (code := high(error$code)) = 7 then do;
266  3               call getpasswd;
267  3               call crlf;
268  3               call setdma(.fcb16);              /* set dma to password */
269  3               curfcb(6) = curfcb(6) or 80h;           /* open in RO mode     */
270  3               call return$errors(0);
271  3               error$code = open$file(.curfcb);
272  3            end;
273  2            else do;
274  3              call print$buf(.failed);
275  3              call print$file(.curfcb);
276  3              call printchar(' ');
277  3              call error(code);
278  3            end;
279  2          if low(error$code) <> 0FFH then
280  2          do;
281  3            call return$errors(0);        /* reset error mode */
282  3            call setdma(.tbuff);
283  3            curfcb(32) = 0;
284  3            eod = 0;
285  3            do while (not eod) and (read$record (.curfcb) = 0);
286  4              do i = 0 to 127;
287  5                if (char := tbuff(i)) = control$z
                       then eod = true;
289  5                if not eod then
290  5                do;
291  6                  if check$con$stat then do;
293  7                    tcnt = conin;
294  7                    call terminate;   /* terminate only on ctrl-c */
295  7                  end;
296  6                  if cnt <> 0 then
297  6                  do;
298  7                    if char = 0ah then
299  7                      call page$test;
300  7                  end;
301  6                  call printchar (char);
302  6                end;
303  5              end;
304  4            end;
               /*  necessary to close under MP/M & Concurrent
                 call close (.curfcb);
               */
305  3          end;
306  2          else call error(0);

307  2      end type$file;

           /*- - - - - - - - - - - - - - - - - - - - - - - - - - - - - -*/

                          /* try typing files one at a time */
308  1      multi$file:
           procedure;
```

COPYRIGHT © 1982
DIGITAL RESEARCH
P.O. BOX 579
PACIFIC GROVE, CA 93950

SER. #

```
309  2          declare (code,dcnt) byte;
310  2          declare (nextfcb$adr,savdcnt,savsearcha,savsearchl) addr;
311  2          declare nextfcb based nextfcb$adr (32) byte;
312  2          declare (hash1,hash2,hash3) address;

313  2          call setdma(.tbuff);
314  2          if (dcnt:=search$first(.fcb)) = 0ffh then
315  2             call error(0);

316  2             do while dcnt <> 0ffh;
317  3             nextfcb$adr = shl(dcnt,5) + .tbuff;
318  3             savdcnt = getscbword(dcnt$offset);
319  3             savsearcha = getscbword(searcha$offset);
320  3             savsearchl = getscbword(searchl$offset);
                   /* save searched fcb's hash code (5 bytes) */
321  3             hash1 = getscbword(hash1$offset);
322  3             hash2 = getscbword(hash2$offset);
323  3             hash3 = getscbword(hash3$offset);   /* saved one extra byte */
324  3             call move(16,nextfcb$adr,.curfcb);  /* copy matched filename */
325  3             curfcb(0) = fcb(0);         /* set drive */
326  3             call page$test;
327  3          call crlf;
328  3             call print$file(.curfcb);
329  3             call printchar(':');
330  3             call page$test;
331  3             call crlf;
332  3             call type$file;
333  3             call setdma(.tbuff);
334  3             call setscbword(dcnt$offset,savdcnt);
335  3             call setscbword(searcha$offset,savsearcha);
336  3             call setscbword(searchl$offset,savsearchl);
                   /* restore hash code */
337  3             call setscbword(hash1$offset,hash1);
338  3             call setscbword(hash2$offset,hash2);
339  3             call setscbword(hash3$offset,hash3);
340  3             if .fcb <> savsearcha then  /*restore orig fcb if destroyed*/
341  3               call move(16,.fcb,savsearcha);
342  3             dcnt = search$next;
343  3             end;
344  2       end multi$file;

             /*- - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -*/

                          /* skip over blanks or tabs in command */
345  1       eat$blanks: procedure;
346  2            do while (m = ' ') or (m = ctrli);
347  3               status = status + 1;
348  3            end;
349  2       end eat$blanks;

             /* - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - */


               /****************************************
               *                                      *
               *      M A I N  P R O G R A M          *
               *                                      *
```

COPYRIGHT © 1982
DIGITAL RESEARCH
P. O. BOX 579
PACIFIC GROVE, CA 93950
SER. #

```
                ***************************************/

350   1       plm:
                do;
351   2           ver = version;
352   2           if (low(ver) < cpmversion) or (high(ver) = mpmproduct) then do;
354   3             call print$buf (.('Requires CP/M 3.0','$'));
355   3             call terminate;
356   3             end;

357   2           call set$console$mode;  /* set program interrupt to control-c only */

                  /* get command */
358   2           if not input$found(.tbuff(1)) then do;
                     /* prompt for file */
360   3              call print$buf(.('Enter file: $'));
361   3              no$chars = read$console$buf(.tbuff(0),40);
362   3              call print$buf(.(cr,lf,'$'));
363   3              tbuff(1) = ' ';  /* blank out nc field */
364   3              tbuff(no$chars+2)=0; /* mark eoln */
                     /* convert input to upper case */
365   3              do i = 2 to no$chars+1;
366   4                 if tbuff(i) >= 'a' then
367   4                    if tbuff(i) < '}' then
368   4                       tbuff(i) = tbuff(i) - 20h;
369   4              end;
370   3           end;

                  /* parse command for file and options */
371   2           tcnt,cnt = 0;
372   2           parse$fn.buff$adr = .tbuff(1);
373   2           parse$fn.fcb$adr = .fcb;
374   2           status = parse(.parse$fn);
375   2           if status = 0FFFFh then do;
377   3              call print$buf(.('ERROR: Invalid file name. $'));
378   3              call terminate;
379   3           end;

                  /* get default paging mode */
380   2           if getscbbyte(con$page$mode) <> 0 then
381   2              paging = false;

                  /* check for options */
382   2           if (status <> 0) then do;   /* options follow? */
384   3             call eat$blanks;
385   3             if m = 0 then goto continue;  /* no options found */
                    /* check for page option */
387   3             if m <> '[' then
388   3                call error$opt(0);
389   3             status = status + 1;
390   3             call eat$blanks;
391   3             if m = 'N' then do;
393   4                status = status + 1;
394   4                if (m = '0') or (m = ' ') then do;
396   5                  status = status + 1;
397   5                  negate = true;
398   5                end;
```

COPYRIGHT © 1982
DIGITAL RESEARCH
P. O. BOX 579
PACIFIC GROVE, CA 93950

SER. #_____

```
                    else
399  4                    call error$opt(1);
400  4                  call eat$blanks;
401  4                end;

402  3                if m = 'P' then
403  3                   paging = true;
                       else
404  3                   call error$opt(1);
405  3                status = status + 1;
406  3                if ( m = ']' ) or ( m = 0 ) then
407  3                   goto continue;
408  3                else if m <> 'A' then
409  3                   goto end$opt;
                       status = status + 1;
411  3                if ( m = ']' ) or ( m = 0 ) then
412  3                   goto continue;
413  3                else if m <> 'G' then
414  3                   goto end$opt;
                       status = status + 1;
416  3                if ( m = ']' ) or ( m = 0 ) then
417  3                   goto continue;
418  3                else if m <> 'E' then
419  3                   goto end$opt;
                       status = status + 1;
421  3                end$opt:
                       call eat$blanks;
422  3                   if ( m <> ']' ) and ( m <> 0 ) then
423  3                      call error$opt(1);
424  3                end;
425  2                continue:

                       /* check for negation of paging */
                       if negate then
426  2                     paging = not paging;

                       /* get page size */
427  2                   if paging then do;
429  3                     cnt = getscbbyte(con$page$size)-2;
430  3                     if cnt = 0 then cnt = 22;      /* by default, 22 lines on screen */
432  3                   end;
                         else
433  2                     cnt = 0;   /* no paging */

                       /* check for wild card or single file */
434  2                    wflag = scanques(.fcb);
435  2                    if wflag = true then
436  2                       call multi$file;
437  2                    else do;
438  3                       call move(16,.fcb,.curfcb);
439  3                       call type$file;
440  3                    end;
441  2                    call terminate;
442  2                 end;
443  1              end type;
```

COPYRIGHT © 1982
DIGITAL RESEARCH
P. O. BOX 579
PACIFIC GROVE, CA 93950

SER. #_____

MODULE INFORMATION:

```
CODE AREA SIZE     = 097DH   2429D
VARIABLE AREA SIZE = 0073H    115D
MAXIMUM STACK SIZE = 000CH     12D
676 LINES READ
0 PROGRAM ERROR(S)
```

END OF PL/M-80 COMPILATION

COPYRIGHT © 1982
DIGITAL RESEARCH
P.O. BOX 579
PACIFIC GROVE, CA 93950

SER. #_____ ____

```
0000 TYPE     0000 TYPE
0000 MCMSOA   0000 MSGA
0004 BDISK    0080 BUFF       0080 BUFFA      0050 CMDRV      0000 CPU
007C CR               006D DOLLA      005C FCB        006C FCB16      005C FCBA
005C IFCB       005C IFCBA      0003 IOBYTE     0053 LEN0       0056 LEN1
0006 MAXB       0006 MEMSIZ     0005 MON1       0005 MON2       0005 MON2A
0005 MON3       0000 OFFSET     006E PARMA      0051 PASS0      0054 PASS1
007F R0               007D RR         007D RRECA      007F RRECO      005C SFCB
0080 TBUFF
0000 TYPE
0D45 MEMORY     028A PLM        0CD2 EOD        0CD3 I          0CD4 CHAR
0CD5 CNT        0CD6 TCNT       0CD7 CODE       0CD8 VER        0CDA ERRORCODE
0CDC PAGING     0CDD NEGATE     0CDE STATUS     0CE0 NOCHARS
0CE1 LASTDSEGBYTE               0CE2 WFLAG      0CE3 CURFCB     0180 MORE
019E FAILED     051C READCONSOLE                0525 PRINTCHAR  0D04 CHAR
0535 CONIN      053E PRINTBUF   0D05 BUFFADR    054E READCONSOLEBUF
0D07 BUFFERADDRESS              0D09 MAX        0575 VERSION
057E CHECKCONSTAT               0587 OPENFILE   0D0A FCBADDRESS 0597 CLOSEFILE
0D0C FCBADDRESS         05A7 READRECORD 0D0E FCBADDRESS 05B7 SETDMA     0D10 DMA
05C7 RETURNERRORS               0D12 MODE       05D7 TERMINATE
05E0 SEARCHFIRST                0D13 FCBADDRESS 05F0 SEARCHNEXT 0D15 SCBPD
05F9 GETSCBBYTE         0D19 OFFSET     0611 GETSCBWORD 0D1A OFFSET     0629 SETSCBWORD
0D1B OFFSET     0D1C VALUE      064F SETCONSOLEMODE             0D1E PARSEFN
0658 CRLF       0663 FILL       0D22 S          0D24 F          0D25 C
068E UCASE      0D26 C          06AB GETPASSWD  0D27 I          0D28 C
06B7 RETRY      06D0 NXTCHR     0736 EXIT       073D ERROR      0D29 CODE
0791 PRINTFILE 0D2A FCBP        0D2C K          0D2D C          07EE ERROROPT
0D2E CODE       0818 INPUTFOUND 0D2F BUFFERADR  084F SCANQUES   0D31 STRADDR
0D33 I          0D34 WILDCARD   0885 PAGETEST   08CA TYPEFILE   09DB MULTIFILE
0D35 CODE       0D36 DCNT       0D37 NEXTFCBADR 0D39 SAVDCNT    0D3B SAVSEARCHA
0D3D SAVSEARCHL         0D3F HASH1      0D41 HASH2      0D43 HASH3      0ADB EATBLANKS
0497 ENDOPT     04B9 CONTINUE
0000 PARSE
0C46 DBLNK1     0C3A DEBLNK     0BF8 DELIM      0C33 DELIM1     0C36 DELIM2
0BF0 PADFLD     0B8E PARS81     0B8F PARS82     0AFD PARSE      0B1E PARSE1
0B4C PARSE2     0B4F PARSE3     0B61 PARSE4     0B65 PARSE5     0B67 PARSE6
0B7B PARSE7     0B7F PARSE8     0BA0 PARSE9     0BC9 PWFLD      0BDC PWFLD1
0BBB SETFD1     0BBD SETFD2     0BC1 SETFD3     0BA6 SETFLD
```

COPYRIGHT © 1982
DIGITAL RESEARCH
P. O. BOX 579
PACIFIC GROVE, CA 93950

SER. #_____

```
0000 TYPE#
0000 MCD80A#
0000 TYPE#
051C 38         051C 39         0525 40         0525 41         0529 43
0534 44         0535 45         0535 46         053E 47         053E 48
0544 50         054D 51         054E 52         0556 56         0560 57
0569 58         0570 59         0575 60         0575 61         0575 62
057E 63         057E 64         057E 65         0587 66         0587 67
058D 69         0597 70         0597 71         059D 73         05A7 74
05A7 75         05AD 77         05B7 78         05B7 79         05BD 81
05C6 82         05C7 83         05CB 85         05D6 86         05D7 87
05D7 88         05DF 89         05E0 90         05E6 92         05F0 93
05F0 94         05F0 95         05F9 96         05F9 98         05FD 100
0603 101  0608 102  0611 103  0611 104  0615 106
061B 107  0620 108  0629 109  0629 110  0631 113
0637 114  063C 115  0646 116  064E 117  064F 118
064F 119  0657 120  0658 125  0658 126  065D 127
0662 128  0663 129  0670 131  067C 132  0683 133
068A 134  068D 135  068E 136  068E 138  0699 139
06A1 140  06A7 141  06AB 142  06AB 143  06AB 145
06AE 146  06B1 147  06B7 148  06C2 149  06D0 150
06DB 151  06E8 152  06F0 153  06F3 154  06FB 155
06FE 156  0706 158  070E 159  0711 161  0721 162
0724 163  0724 164  0724 165  072C 166  072F 167
0736 168  073C 169  073D 170  0741 172  0749 174
074F 175  0752 176  0752 177  075A 179  0760 180
0763 181  0763 182  076B 184  0771 185  0774 186
0774 187  077C 188  0782 189  078A 190  0790 191
0791 192  0797 197  07A0 199  07AA 200  07AF 201
07AF 202  07BD 203  07C5 204  07CA 205  07DF 206
07E6 207  07ED 208  07EE 209  07F2 211  07F8 212
0800 213  0806 214  080E 215  0814 216  0817 217
0818 218  081E 221  0835 222  083C 223  083F 224
0848 225  084B 226  084E 227  084F 228  0855 233
085A 234  085D 235  0865 236  086E 237  0873 238
0877 239  087E 240  0881 241  0885 242  0885 243
0885 244  088D 245  089B 247  08A1 248  08A7 249
08AC 250  08B4 251  08B7 252  08BF 253  08C4 254
08C9 255  08C9 256  08CA 257  08CA 258  08CF 259
08D5 260  08DD 261  08E2 262  08EB 263  08F4 264
0900 266  0903 267  0906 268  090C 269  0914 270
0919 271  0922 272  0925 274  092B 275  0931 276
0936 277  093D 278  093D 279  0946 281  094B 282
0951 283  0956 284  095B 285  0972 286  0980 287
0992 288  0997 289  099E 291  09A5 293  09AB 294
09AE 295  09AE 296  09B6 298  09BE 299  09C1 300
09C1 301  09C8 302  09C8 303  09CF 304  09D2 305
09D5 306  09DA 307  09DB 308  09DB 313  09E1 314
09EF 315  09F4 316  09FC 317  0A0E 318  0A16 319
0A1E 320  0A26 321  0A2E 322  0A36 323  0A3E 324
0A52 325  0A58 326  0A5B 327  0A5E 328  0A64 329
0A69 330  0A6C 331  0A6F 332  0A72 333  0A78 334
0A81 335  0A8A 336  0A93 337  0A9C 338  0AA5 339
0AAE 340  0ABB 341  0ACE 342  0AD4 343  0AD7 344
0AD8 345  0AD8 346  0AF2 347  0AF9 348  0AFC 349
0287 350  028D 351  0293 352  02AB 354  02B1 355
02B4 356  02B4 357  02B7 358  02C2 360  02C8 361
02D3 362  02D9 363  02DE 364  02E9 365  02F9 366
0308 367  0317 368  032A 369  0334 370  0334 371
033C 372  0342 373  0348 374  0351 375  035E 377
0364 378  0367 379  0367 380  0371 381  0376 382
0382 384  0385 385  038E 386  0391 387  039A 388
039F 389  03A6 390  03A9 391  03B2 393  03B9 394
```

COPYRIGHT © 1982
DIGITAL RESEARCH
P. O. BOX 579
PACIFIC GROVE, CA 93950

SER. #_____

```
03D3 396    03DA 397    03DF 398    03E2 399    03E7 400
03EA 401    03EA 402    03F3 403    03FB 404    0400 405
0407 406    0421 407    0424 408    042D 409    0430 410
0437 411    0451 412    0454 413    045D 414    0460 415
0467 416    0481 417    0484 418    048D 419    0490 420
0497 421    049A 422    04B4 423    04B9 424    04B9 425
04C0 426    04C7 427    04CE 429    04D8 430    04E0 431
04E5 432    04E8 433    04ED 434    04F6 435    04FE 436
0504 438    0514 439    0517 440    0517 441    051A 443
0000 PARSE#
```

COPYRIGHT © 1982
DIGITAL RESEARCH
P.O. BOX 579
PACIFIC GROVE, CA 93950

SER. #_____